Text Summarization

By: Team 12

Isha Arora

Akshay Chopra

Shivalik Narad

Preksha Raj Shimoga Basavaraja

INTRODUCTION

Too long; didn't read (**tl;dr**)

For the long articles and blog posts which you are interested in but sometimes you just need a gist

Here we present two different algorithms to create a summary for the long text that you want to read!

Summarizers Implemented:
Text-rank
Lex-rank

DATA COLLECTION

Data is collected by the following sources-
https://github.com/kylehg/summarizer/tree/master/input

The corpus is a transcript of news articles collected from APW and New York Times

After the data was collected we preprocessed the data in the following way

- cleaned the data
- removed the trailing white spaces
- split the data into lines
- removed the punctuations and numbers from each line
- tokenized the lines
- then split the links into tokens and stemmed the words-used for 2nd approach.

TEXTRANK

- The input file is preprocessed to return a list of sentences
- Based on this list of sentences create a graph
- In the Graph each node is marked with each sentence
- The similarity between each sentences is calculated and is assigned as weights of edges between the nodes
- The page rank is applied to each node to get the rank of each node (sentence)
- The topmost rank sentences is returned as the summary

SECOND APPROACH

The first approach used gave the summary based on the topmost ranked sentences

We also worked an approach to give the topmost keywords that summarize the article.

The input lines is split into tokens and tokens are assigned as nodes.

Then pagerank is applied to this graph to get the top ranked keywords.

The top ranked keywords are returned as the keyword summary.

Summary can also be extracted as a percentage of the article size.

LEX RANK

Text Rank implements page rank algorithm but it doesn’t take into consideration the how important is a word!

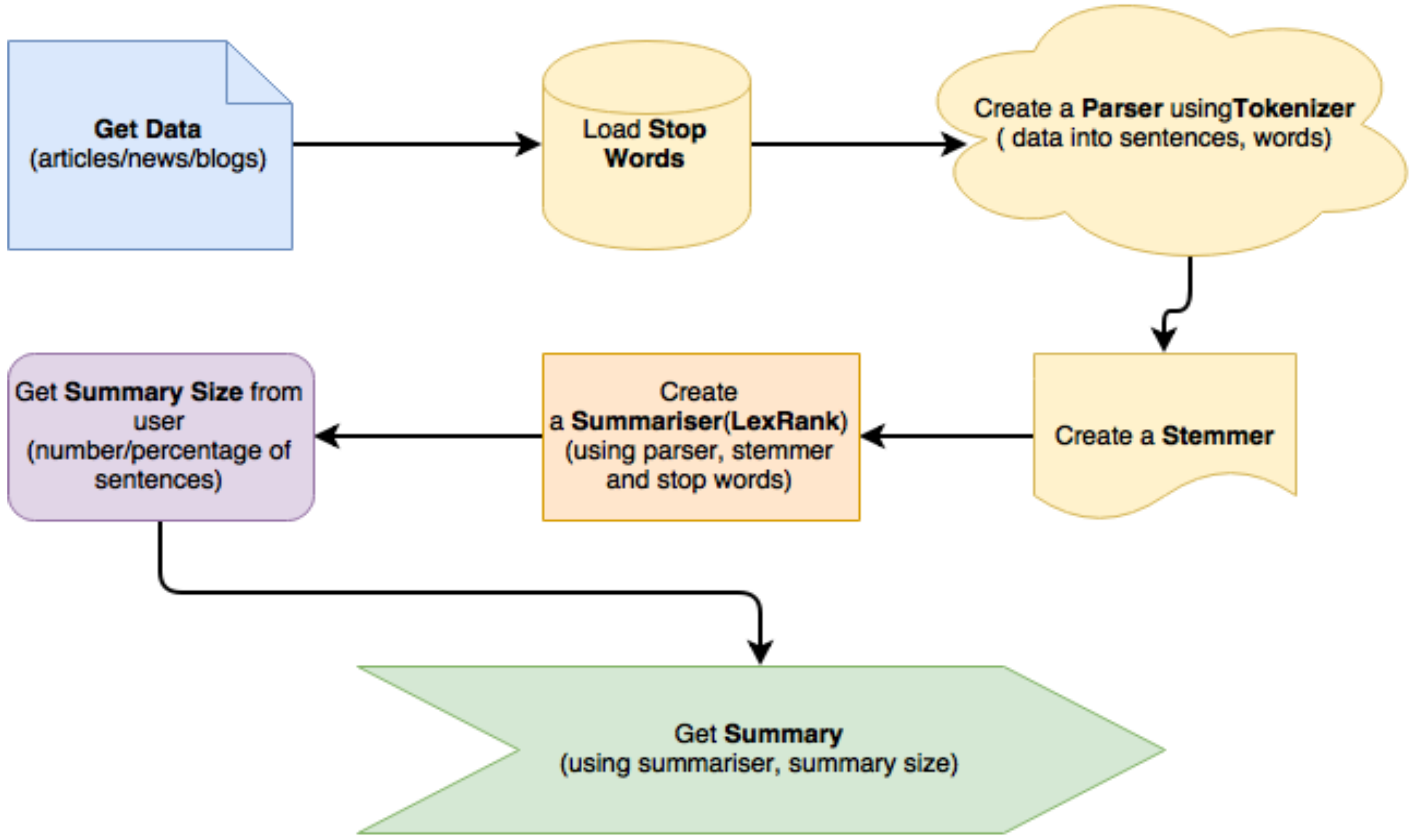
Lex rank features:

- Ranks the sentences based on words that it contains
- Word importance taken into consideration while ranking (tf-idf scores)
- Lex-Rank algorithm can work for multi-document summarisation

LEX RANK Contd..

- Preprocessing required:
  a. Sentence extraction from paragraphs
  b. Tokenization of sentences to words
  c. Stemming the words
  d. Stop words elimination
- Calculate tf-idf metrics for words in the document corpus
- Rank the sentences using the above metrics
- Produce summary based on the sentence ranking

LEX RANK Simplified


Get Data
(articles/news/blogs)
Load Stop Words
Create a Parser using Tokenizer
(data into sentences, words)
Create a Stemmer
Create a Summariser(LexRank)
(using parser, stemmer and stop words)
Get Summary Size from user
(number/percentage of sentences)
Get Summary
(using summariser, summary size)

REFERENCE SUMMARIES

Developed the script to get other reference summary for comparison.

- Made an API call to the python package called SUMY to retrieve summaries based on sentence count.
- Got the summary of SUMY-Textrank and SUMY-LexRank giving the filename as the input
- Specify the sentence count
- These summaries are considered as the reference summary to compare.

EVALUATE RESULTS-ROGUE

- The sentences retrieved from the SUMY API text-rank is compared with the our text-rank implementation.
- The sentences retrieved from the SUMY API lex-rank is compared with the our lex-rank implementation.
- Also the summaries generated by tex-trank and lex-rank are compared to evaluate which algorithm is efficient for text summarization based on rogue metrics.

thank you!